

beatRunner Mobile App

ECE4007 Senior Design Project

Section L05, beatRunner Team
Project Advisor, Dr. Arthur Koblasz

Oren Levy, Team Leader
Marc Smith
Marvin Howard
Lakshmi Baddam

Submitted

March 5, 2011

Table of Contents

Executive Summary.....	iii
1. Introduction.....	1
1.1 Objective.....	1
1.2 Motivation.....	2
2. Project Description and Goals.....	2
2.1. Problems and solutions: Heart Rate Data Transmission.....	3
2.2. Music Manipulation.....	5
2.2 Instantaneous Workout Data and Workout Summary.....	7
3. Technical Specification.....	8
3.1. Audio Specifications.....	8
3.2 Application Specifications.....	9
4. Design Approach and Details.....	9
4.1 Design Approach.....	9
4.1.1 iOS SDK.....	10
4.1.2 Graphical User Interface.....	11
4.1.3 Target Heart Rate and Calories Burned.....	13
4.1.4 Special Cases: Notifications and Interrupts.....	14
4.2 Codes and Standards.....	15
5. Schedule, Tasks, and Milestones.....	16
6. Results and Acceptance Testing.....	17
7. Budget and Cost Analysis.....	17
8. Conclusions and Future Work.....	18
References.....	19
Appendix 1.....	21

Executive Summary

beatRunner is an iOS operating system compatible workout tool. It serves as a workout organizer and performance view tool that allows the user to analyze and compare his/her workout performance. The app provides the user with three workout options: cardio, fat burning and cross training. The user may customize the intensity of the workout and time durations. He/she may also select a customized playlist to play during his/her workout. All workouts completed can be saved by the user in the iOS device. Each workout's summary includes the date, time, distance, calories burned, and the target BPM. The beatRunner app requires the user to own a Polar T31 coded heart rate monitor and Sparkfun's heart rate monitor interface in order to retrieve the user's heart rate data. A simple music selection algorithm was implemented to allow the music player to choose songs that are closest the user's target BPM. The beatRunner app is required to be submitted for review to Apple Inc before being sold in the 'App Store'. Once Apple Inc approves it, the app will be published in the 'App Store' and will be available to all iOS mobile device users to download at the cost of one dollar per download. The demonstration of the final product was held in the Van Leer building. The final product's main features were illustrated during the presentation such as updating the user's heart rate to the iOS device, choosing songs according to the the user's target BPM, and saving the user's workout results in the iOS device.

1 INTRODUCTION

The design team spent \$184.40 of funding to design the beatRunner app. The beatRunner app is a workout motivational workout app that serves as an organizational tool for users. The beatRunner app is compatible with iOS based mobile devices and provides three workout options for the user.

1.1 Objective

Motivational App

The main objective was to create a motivational app that will encourage users to reach their target heart rates during their workouts. The beatRunner app was originally intended to change the tempo of the music in the user's iPod playlist in order to motivate the runner to reach his/her target heart rate. Although the music manipulation algorithm was designed, it was concluded that it was too slow for real time use as the iOS mobile device took 40 seconds to analyze and change the tempo of a particular song.

Performance Viewer and Workout Organizer App

All workouts saved and completed using the beatRunner app are available for the user to view in his/her iOS device. Workout statistics such as workout type, time, distance, calories burned, and target BPM during a particular workout are saved. The user may retrieve each workout's information by date to analyze and compare his/her workout performance.

User-friendly App

The beatRunner app provides easy navigation features, minimizes user error, and provides customization ability to users. Easy navigability between tabbed windows in the app is accomplished using navigation keys.

1.2 Motivation

The beatRunner app is a new standalone design and is not based on a previous project. Similar to other apps, the beatRunner app serves as a workout organizational tool that allows the users to retrieve workout history to evaluate their performance over time. Unlike these apps, the beatRunner app uses a music selection algorithm to encourage the user to achieve his/her target heart rate. The target user for the beatRunner app is an iOS mobile user who:

- Owns a Polar Coded T31 heart rate monitor (Polar T31 HRM)
- Uses music as a motivator to workout
- Needs a workout organization tool

2 PROJECT DESCRIPTION AND GOALS

The beatRunner app serves as a workout tracking and motivating tool for users. Table 1 below contains the list of planned and actual features.

Table 1 – Planned Features vs Actual Features

Planned Features	Actual Features
Portable workout tool	Non – portable workout tool
HRM data transmission via Bluetooth	HRM data transmission via USB and Internet
Music Manipulation • Speeds up and slows down songs	Music Manipulation • Uses song selection algorithm
Provides instantaneous workout data during workout	Provides instantaneous workout data during workout
Provides workout summary history	Provides workout summary history

The actual product is not portable as planned. It is a consequence of eliminating Bluetooth from the project and transmitting heart rate data via USB to a PC and sending the data from the PC to the iOS device wirelessly. Explained below are the approaches taken for problems and how a final solution was achieved.

2.1 Problems and Solution: Heart Rate Data Transmission

Problems Encountered with Polar WL HRM

The initial goal was to pair the Bluetooth enabled HRM to the iOS device, transmit the user's heart rate via Bluetooth, and receive and display the data on the iOS device. In order to test the functionality of the Polar WL HRM, it was paired with a Mac Book Pro laptop. CoolTerm, a serial port terminal application was utilized to receive packets and display the data. In order to successfully receive the transmitted data, information about the Polar WL HRM such as baud rate, parity, number of bits, number of stop bits is required. The user manual did not consist of the necessary information and a Polar support personal was contacted regarding the issue.

However, personal informed the team that the information is proprietary. The team used a trial and error method by experimenting with different baud rates, parities, number of bits and stop bits. However, it was determined that the received packets from the Polar WL HRM were specially encoded, and it requires a specific command to decode the data.

Problems Encountered with Bluetooth and iOS

An alternative method was chosen to extract readable heart rate data from a heart rate monitor and transmit data to an iOS device. It required the use of the following components:

- Polar Coded T31 HRM to provide the user's ECG signal
- Heart Rate Monitor Interface (HRMI) to convert the ECG signal to readable heart rate data
- Arduino BT microcontroller to receive data from HRMI and transmit to iOS device via Bluetooth

Important Bluetooth information such as baud rate, parity, number of bits, number of stop bits was provided in HRMI data sheet. In order to test the functionality of the module, the Arduino BT was successfully paired with a MacBook Pro initially and heart rate data was displayed in CoolTerm. However, the pairing of the Arduino BT to an iOS device was unsuccessful. It was determined using research that the pairing with an iOS device is limited to Apple approved audio accessories. In addition, during the research it was found that there is no functionality for iOS apps to transfer or receive data and files via Bluetooth. The Arduino BT is not an Apple approved Bluetooth device, and the app requires data transfer, so it was concluded that Apple's native Bluetooth stack is not useful to successfully receive heart rate data on an iOS device. An alternative to Apple's native Bluetooth stack called BTstack is available. The pros and cons of the BTstack are provided in Table 2.

Table 2 - BTStack

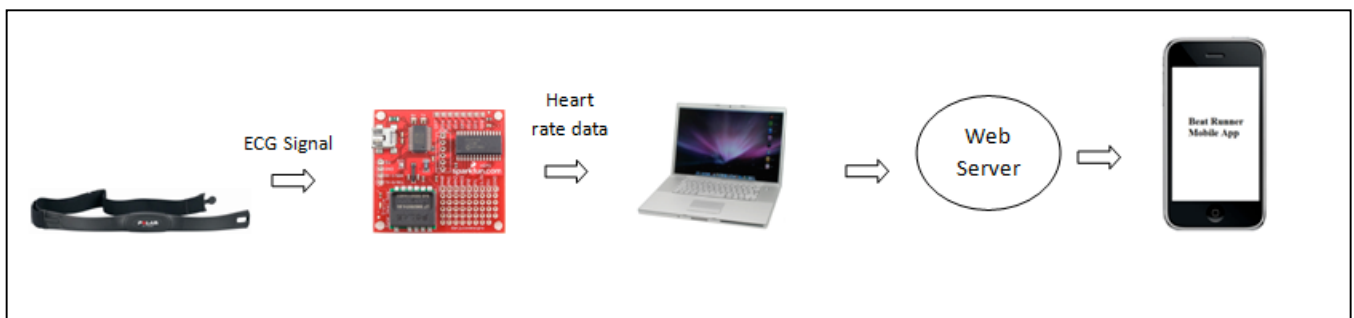
Pros	Cons
Unrestricted pairing	iOS mobile device jailbreak required
Data and file transfer	Solid understanding of Bluetooth protocol required
	Not tested or compatible with latest iOS (4.3)

Although the BTstack provides unrestricted pairing and data receiving/ transmitting functionality via Bluetooth, an iOS device is required to be jailbroken in order to override Apple's Bluetooth stack and download the BTstack. This is a major disadvantage, as Apple requires all apps to be functional on non-jailbroken devices in order for it to be sold in the App Store. Also, there is a steep learning curve for a developer to implement the BTstack in his/her application. The BTstack architecture consists of a Bluetooth server that contains a run loop, socket server and the

Bluetooth stack [11]. It is important for the developer to understand how the data packets are sent, received and extracted at each stage in order to successfully implement the BTstack in his/her application. It was beyond the scope of the project to do deep research on Bluetooth protocol, therefore the use of Bluetooth to transmit heart rate data was eliminated.

Internet based Solution to Transmit Heart Rate Data

Figure 1 – Design Data Flow



Since the use of Bluetooth was unsuccessful in transmitting data to the iOS device, the final product uses the Internet to transmit heart rate data to the iPhone. Figure 1 depicts the data flow of the final design. The user's ECG signal is transmitted from Polar T31 HRM and received by the HRMI via a magnetic field. Then the readable heart rate data is transferred to a PC via USB. Then an executable that using the C programming language receives and verifies the data from the PC's USB port and posts the data to a PHP script stored on the web server. The beatRunner app then reads the data from the web page and displays it on the iOS device.

2.2 Music Manipulation Feature

Tempo Changing Algorithm

The original music manipulation feature consisted of two components as shown in Table 1. The tempo-changing feature was designed and tested to slow down and speed up songs. The Dirac

library for iOS was used to control the tempo of a song while preserving the pitch. However, each song consumed 40 seconds of processing time to change the tempo. As a result, it was determined that the music manipulation algorithm was too slow to implement in the beatRunner app where the song must be played instantly when picked.

Alternative: Changing Playback rate

An alternative to changing the tempo while preserving the pitch is to change the playback rate by altering both the pitch and the tempo. An Apple-provided library called MediaPlayer was used in order to change a song's playback rate in real-time. A property called `currentPlaybackRate` is listed in the class's reference to manipulate a song's playback rate. Although the class provides a description of the property, when used it resulted in a crash in the program. After researching through developer forums, it was determined that the property is not accessible to developers as advertised in the class reference. It can only be altered if the developer has access to the .m files of the Media Player library. However, the .m files are closed-source and Apple does not give access to developers view or download them.

After two unsuccessful attempts at devising algorithms to implement music manipulation in the application, it was determined that this segment of the application was beyond the scope of the project. To allow for real-time music manipulation, each song would need to be split into small frames. Each frame would need to be windowed, processed, and then reconstructed for playback. Currently, there are not any music manipulation libraries available for iOS which provide this method of processing. A new library would have to be written in order to achieve this, which is a project in its own right. Therefore, it was concluded to remove the manipulation feature from the final product.

A song selection algorithm was implemented instead. Given that the user selects a playlist of song's the algorithm extracts each song's BPM and compares the BPM with the user's target BPM. If the song's BPM is within $\pm 20\%$ range of the target BPM, the song is selected to play to encourage to the user to run at the BPM's pace. Otherwise, the range is incremented by 5% until a song is found. Using the algorithm, the music player will choose songs that are closest the user's target BPM.

2.3 Instantaneous Workout Data and Workout Summary

The final product provides the user instantaneous workout data:

- Elapsed time
- Distance ran
- Current heart rate
- Target heart rate
- Calories burned

The initial prototype consisted of a Polar WL HRM and an iOS mobile device. The final prototype has been altered due to problems encountered during the design with Bluetooth and music manipulation.

Table 3 depicts the information about the planned prototype and the actual prototype built.

Table 3 – Planned vs Actual Prototype Parts

Planned Parts	Actual Parts
Polar WearLink + Transmitter Bluetooth heart rate monitor (Polar WL HRM)	Polar T31 HRM
iOS mobile device	iOS mobile device
	MacBook Pro laptop
	Heart Rate Monitor Interface
	USB cable

3 TECHNICAL SPECIFICATIONS

3.1 Audio Specifications

Table 4 lists the audio specifications for the BeatRunner app. The audio sampling frequency was set to 44.1 kHz, which the industry standard for audio is encoding. Audio is played back in stereo, meaning the user can listen to two independent audio channels. In order to provide a smooth transition between songs for the user, the time latency between songs will be 0 sec. All audio specifications were met in the final product as the features were not altered in the design process.

Table 4 - Audio Specifications

Feature	Specification	Met
Audio Sampling Frequency	44.1 kHz	Yes
Audio Output	Stereo (2 Channel)	Yes
Latency between songs	0 sec	Yes

3.2 Application Specifications

Table 5 describes Apple's hardware specifications for the iOS operating system. The three-axis accelerometer works as a pedometer and is used to calculate the distance ran in a workout. There are four push buttons used: two push buttons to control the app's audio volume, one push button to navigate back to the home screen from the app, and one to lock the device while the app is running. The Multi Touch Screen Display is required for the user to communicate with the app using touch gestures. The beatRunner app was tested to compatible with mobile devices running on iOS operating system 4.3 or later. The initial design was not compatible with iOS versions before 4.3 as the app utilizes the BPM tag feature only available after 4.3. This feature is used to choose choice and categorize songs according to BPM.

Table 5 - Application Specifications

Feature	Specification	Met?
Used iPhone inputs	3 axis accelerometer Multi Touch Screen Display 4 Push Buttons	Yes
Supported operating systems	iOS 2.0 or later	iOS 4.3 or later
Compatible Mobile Devices	iPhone (Original,3G,3GS,4) iPod Touch (2nd generation and later) iPad	Yes

4 DESIGN APPROACH AND DETAILS

4.1 Design Approach

As originally planned, the beatRunner app was built using the iOS software development kit (SDK) developed by Apple. The iOS SDK contains frameworks and Xcode tools that will be utilized in the beatRunner app design.

4.1.1 iOS SDK

Frameworks

As originally planned Layers from the iOS SDK were used to implement functionalities. The Cocoa Touch Layer's UI Kit was used for event handling. For example, when the user clicks the 'Done' button depicted in Figure 4, the UI Event class in the UI Kit is utilized to detect the click and launch the Current Workout Status screen displayed in Figure 5. The Media Player framework in the Media Layer was used to access to the user's iPod library in the mobile device and support for playing tracks and play lists. It was utilized to allow the user to select a customized play list before a workout (see Figure 4) and pause, skip and play songs during his/her workout (see Figure 5). To satisfy the task of pedometer, the beatRunner application uses a function within the Shake API. The Shake API was used to detect shakes in the iPhone. There are three events that are used to implement this method: Motion Begin, Motion End, and Motion Cancelled. These three events allow the application to account for every movement while the code is being executed. As soon as the device is moved, the Motion Begin increments the count by two, (two to account for each step of the individual) and calls Motion End after each step. If the application is exited out of or phone completely stops movement then Motion Cancelled is called causing the method to end.

Tools

Three Xcode tools were utilized to design, create, debug, and optimize the beatRunner app. The beatRunner app code was written in Objective-C and was created and compiled in the Xcode integrated development environment (IDE) tool. As originally planned, The IDE was used in conjunction with the Simulator tool to test the app's user interface (UI) prior to uploading to a mobile device. The Interface Builder (IB) was utilized to customize and create the app's screens,

depicted in Figures 2, 3, and 4 visually. GUI components such as buttons, navigation keys, labels and drop down menus were constructed and customized using IB as planned.

4.1.2 Graphical User Interface



Figure 2. New Workout Screens

The New Workout Screen is displayed when the user loads the beatRunner app on the mobile device and is depicted in Figure 2. The user will select his/her workout type from the following 3 categories: Cardio, Fat burning, or Cross training. If the user does not choose the workout, the default workout type is set to Cardio. The level selection is used to determine the user preferred “easiness” of the workout. This selection and the user’s age determine the user’s target BPM for the workout using the formula:

$$\text{Target BPM} = 220 - \text{Age} - \text{Level}$$

Equation 1

A level of Easy, Medium, and Hard is set to 50, 35, and 15 respectively. For example, if a user’s age is 20 and chooses a level of Hard level workout. Then, the music selection algorithm will choose songs with BPM closest to 185 BPM.

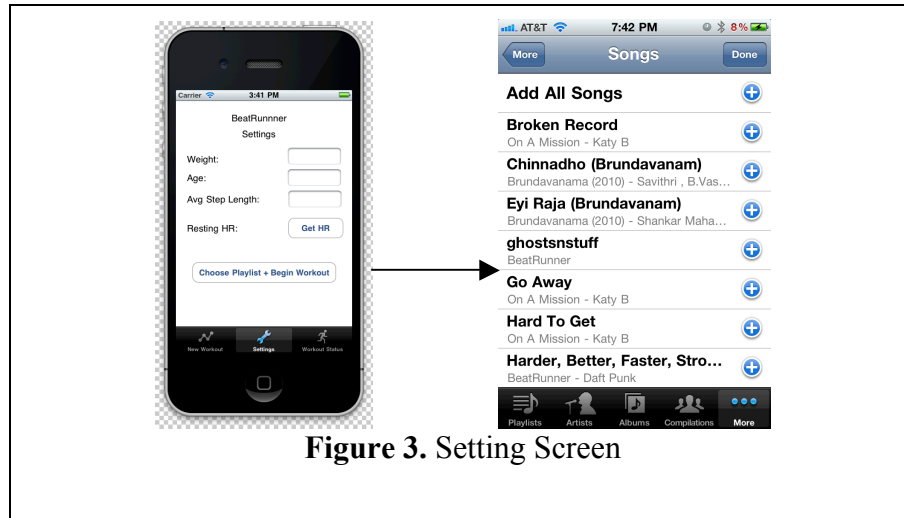


Figure 3. Setting Screen

Before the user begins the workout, he/she must update the Settings Screen shown in Figure 3. Next, the user may select a customized play list from his/her iPod music folder or play all the songs on his/her iPod. This screen contains user defined variables such as weight, age and resting heart rate which are required to calculate valuable statistics for the app such as the target heart rate. The user may choose to input his/her average foot step length in feet by filling out the respective text field in the Settings screen. If the user chooses not to enter his/her average foot step length, then the default average value of two feet per step is used to calculate the distance ran by the user. Once the user is ready to calculate his/her resting heart rate, he/she is required to press the 'Get HR' button. The resting heart rate is used to calculate the user's target heart rate in the Workout Status screen depicted in Figure 4. As soon as the 'Create Playlist + Begin Workout' button is clicked, the user is navigated to his/her iPod library to choose songs. Once the user clicks the 'Done' button in the iPod library screen, the user will be able to view his/her current work out details in the Workout Status screen depicted in Figure 4.

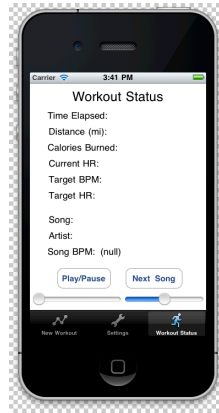


Figure 4. Workout Status Screen

The details in the Workout Status screen include the amount of time since the workout was started, the distance ran, current heart rate, target heart rate, target BPM, and calories burned. The application will display the song currently playing, and will allow the user to play/pause the workout or skip to the next song. Once the duration of the workout is reached a ‘Save’ button at the upper left corner is set visible in order to allow the user to save each workout’s statistics.

4.1.3 *Target Heart Rate and Calories Burned*

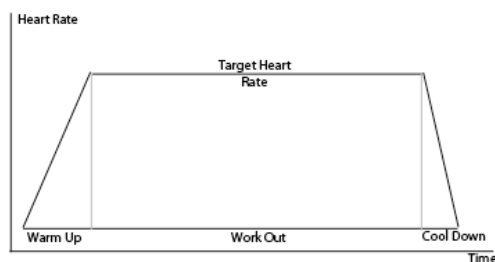


Figure 5. Example of a full workout.

As originally planned, the final beatRunner app require the user to input his/her age, resting heart rate, and choose from three workouts. All three workouts take the shape of the graph in Figure 5 where the x axis is time and the y is heart rate. The only difference between the three workouts is the target heart rate (THR) zone. The maximum heart rate (MHR) for the user will be determined by the app by subtracting the user's age from 220 [2]. The target heart rate for each workout is calculated using the Karvonen method. The Karvonen method uses the resting heart rate (RHR) to find the THR using the Equation 2.

$$\text{THR} = (\text{MHR} - \text{RHR}) * \text{intensity} + \text{RHR}$$

Equation 2.

The THR zone is found by obtaining the THR for the upper and lower percentage of the workout. For a fat burning workout, the target heart rate zone is between 60% and 70% intensity. A cardio workout has a THR zone between 70% and 80% intensity. A cross-training workout has a THR zone between 80% and 85% intensity.

The calories burned during a workout were determined using Equation 3 considering the user's age in years, weight in pounds, average heart rate (AHR) and elapsed workout time [2].

$$\text{Calories Burned} = (0.2017 * \text{age} + 0.09036 * \text{weight} + 0.6309 * \text{AHR} - 55.0969) * \text{elapsed time} / 4.184$$

Equation 3.

4.1.4 Special Cases: Notifications and Interrupts

Software Interrupts

Software Interrupts were designed as originally planned. If the user decides to either answer, decline or ignore a phone call while the app is running, the workout and songs are both paused and automatically restarted when the call is ended. When a text message is received the

app does not be affected.

Hardware Interrupts

Planned hardware interrupts were successfully implemented in the final design. When the app is running and the user presses the lock or home key, the workout and music continues uninterrupted.

Battery Outage

In case of a battery outage during a user's workout, all current workout statistics will be lost and will not be saved.

4.2 Codes and Standards

The beatRunner was designed to work on the Apple's iOS operating system. The app was written in Objective-C and was created in the Xcode integrated development environment (IDE). The design was altered due to Apple's Bluetooth restrictions on data transmission to use the Internet to transmit the heart rate data. Heart rate data is required to be transferred to the PC using a USB prior to posting the heart rate wirelessly to a web server thereby making the app not portable. The limited processing power of iOS restricted us from implementing the tempo changing music manipulation algorithm originally designed. The app was developed using frameworks and resources provided by the OS SDK for developers. The beatRunner app has to be approved by Apple Inc before marketing it in the 'App Store'. It is required to be submitted as a fully functional app, and cannot be submitted as a trial, demo or beta version app.

5. Scheduled Tasks and Milestones

Over the last three months, several steps were vital to the completion of the beatRunner app. Table 6 depicts the major milestones and tasks to be accomplished in italics and bold print respectively.

The programming process will begin upon conclusion of the proposal. The beatRunner project will be completed in modules by dividing each task among the team members to reduce redundancy and utilize time efficiently. The first step will begin with the construction of the user interface. Afterwards, the programming of the heart rate monitor, accelerometer, and the music manipulation will commence simultaneously. Once completed, all modules will be combined for integration and simulation. The final product produced will be demonstrated and presented at the end of the semester. Appendix 1 provides a more detail account of responsibilities and difficulties as well as a Gantt chart showing the overall schedule of events.

Table 6 – Schedule of Task and Milestones

Task Name	Duration	Start	End
Proposal	6 Days	2/14/11	2/21/11
<i>Turn in Proposal</i>	<i>0 Days</i>	2/21/11	2/21/11
User Interface	60+ Days	2/21/11	5/4/11
<i>Construct Screens (GUI)</i>	<i>0 Days</i>	3/8/11	3/8/11
Heart Rate Monitor	45+ Days	3/9/11	4/25/11
<i>Interfacing w/ iPhone</i>	<i>0 Days</i>	3/11/11	3/11/11
<i>Alternative Solution</i>	<i>0 Days</i>	3/15/11	3/15/11
Accelerometer	16 Days	4/9/11	4/29/11
<i>Develop Pedometer Algorithm</i>	<i>0 Days</i>	4/15/11	4/15/11
Music Manipulation	50 Days	3/9/11	4/30/11
<i>Develop Tempo Adjustment Algorithm</i>	<i>0 Days</i>	4/5/11	4/5/11
Integration	15 Days	4/5/11	4/20/11
<i>Combine Modules</i>	<i>0 Days</i>	4/13/11	4/13/11
Demonstration	6 Days	4/29/11	5/6/11
<i>Demo</i>	<i>0 Days</i>	5/5/11	5/5/11
FINAL PRESENTATION	3 Days	4/17/11	4/20/11

6. RESULTS AND ACCEPTANCE TESTING

Testing was done to determine if the use of the accelerometer's shake API resulted in accurate step detection. For a five-minute run, 90% of the steps were detected which was sufficient. The user interface navigability was also tested to determine the average time a user will spend between starting the app and actually beginning a new workout. Before the user begins the workout he/she must choose the workout type, level and duration, complete the user specific information such as weight etc. in the Settings screen and choose a playlist. The goal, which was to build an easily navigable user interface, was achieved as it took less than five seconds to navigate between the Settings screen and to begin the actual workout. The accuracy of heart rate data was also tested. The HRMI transmits 32 bit data in which the first byte indicates whether accurate data was transmitted. A value of one indicates accurate data and all other values indicate spurious data. For total of 100 readings, 93% of the readings were detected to be accurate which is sufficient. During the demonstration the tested features were displayed in the app and successfully verified.

7. BUDGET AND COST ANALYSIS

The total development cost for a prototype of the beatRunner app was \$56,908.22. The equipment cost for the prototype was higher than originally planned since the Polar WL HRM was replaced with the Polar T31 HRM and the Heart Rate Monitor Interface (HRMI). The additional cost for the Polar T31 and the HRMI was \$44.50 and \$59.95 respectively. The hourly

labor wage was set to \$41.27 and takes fringe benefits and overhead costs into account; the total labor cost amounts to \$56,828.27. As planned the music manipulation module required higher number of labor hours than others. However, problems with Bluetooth communication was predicted to take less than 24 labor hours but the resultant design required the team to work more than 60 labor hours on debugging and researching on methods to establish Bluetooth connectivity with the iOS device.

Table 7 - Parts

Part	Price (\$)
Polar T31 HRM	44.50
Polar WL HRM (not used)	59.95
HRMI	79.95

8. Conclusions and Future Work

The demonstration of the altered designed was successful as it verified the app features and heart rate readings. The two main issues in the project were the inability to use Bluetooth to transfer heart rate data to the iOS mobile device and implementing the music manipulation algorithm. It will be a challenge for future engineers to implement a music manipulation algorithm in the app as mobiles devices have low processing power and changing the tempo of a song requires complex DSP algorithms. Future work should be devoted to making the app portable by developing it for another mobile environment such as Android that utilizes an open source operating system and gives the developer the freedom of using Bluetooth to transfer data. This would require developing the app from scratch, as Android developers are required to use the

Java programming language to develop apps.

REFERENCES

- [1] E. Quinn. (2010, May 14). *The Warm Up - How To Warm Up Before Exercise* [Online]. Available: <http://sportsmedicine.about.com/cs/injuryprevention/a/aa071001a.htm> [Accessed February 4, 2011]
- [2] M. Mulrooney. (2010, January 31). *How to Calculate Heart Rate With the Karvonen Method* [Online]. Available: <http://www.livestrong.com/article/78034-calculate-heart-rate-karvonen-method/> [Accessed February 20, 2011]
- [3] K. Flegal. (2010, January 13). *Prevalence and Trends in Obesity Among US Adults, 1999-2008* [Online]. Available: <http://jama.ama-assn.org/content/303/3/235.full?ijkey=ijKHq6YbJn3Oo&keytype=ref&siteid=amajnl> [Accessed February 18, 2011]
- [4] B. Dolan. (2011, January 20). *Study: 42 percent of U.S. uses a smartphone* [Online]. Available: <http://mobihealthnews.com/6178/study-42-percent-of-u-s-uses-a-smartphone/> [Accessed February 18, 2011]
- [5] Nike. *Nike+* [Online]. Available: <http://www.nikeplus.com> [Accessed February 18, 2011]
- [6] Apple. *Nike BOOM* [Online]. Available: <http://itunes.apple.com/us/app/nike-boom/id381439239?mt=8> [Accessed February 18, 2011]
- [7] A. Viklund. *Workout Music Studio: Unleash the Power of your Music!* [Online]. Available: <http://www.bestworkoutmusic.com/repacer.html> [Accessed February 19, 2011]
- [8] Apple Inc., “Library,” [Online]. Available: <http://developer.apple.com/library>. [Accessed: Feb 05, 2011].

- [9] Listening To Music When You Work Out Helps Your Fat Loss., Manvfitness, [Online]. 2010, <http://www.manvfitness.com/559/listening-to-music-makes-you-fitter-helps-lose-body-fat-what/> [Accessed February 20, 2011]
- [10] Are Americans Exercising Less?, WebMD. [Online] Available: <http://www.webmd.com/fitness-exercise/news/20100121/are-americans-backing-off-exercise> [Accessed February 20, 2011]
- [11] M. Ringwald.(2011, April 12).btstack:Architechure (Revision 1090) [Online]. Available: <http://code.google.com/p/btstack/wiki/Architecture>

Appendix 1 - Table A1 – Gantt Chart Details

Task Name	Duration	Start	End	Difficulty	Responsible
Proposal	6 Days	2/14/11	2/21/11		
<i>Turn in Proposal</i>	<i>0 Days</i>	2/21/11	2/21/11	Milestone	Team
User Interface	60+ Days	2/21/11	5/4/11		
<i>Construct Screens(GUI)</i>	<i>3 Days</i>	2/21/11	2/23/11	Milestone	Team
<i>Develop Screen Navigation</i>	<i>6 Days</i>	2/24/11	3/3/11	Medium	Team
<i>Functionality</i>	<i>9 Days</i>	2/24/11	3/8/11	Hard	Lakshmi
Heart Rate Monitor	45 Days	3/9/11	4/25/11		
<i>Connect w/ iPhone</i>	<i>3 Days</i>	3/9/11	3/11/11	Milestone	Marc
<i>Alternative Solution</i>	<i>3 Days</i>	3/12/11	3/15/11	Milestone	Marc
<i>Display data in GUI</i>	<i>3 Days</i>	3/16/11	3/18/11	Medium	Lakshmi
Accelerometer	16 Days	3/9/11	3/29/11		
<i>Develop Pedometer Algorithm</i>	<i>6 Days</i>	3/9/11	3/15/11	Milestone	Oren
<i>Display info in GUI</i>	<i>10 Days</i>	3/16/11	3/29/11	Medium	Lakshmi
Music Manipulation	50 Days	3/9/11	4/30/11		
<i>Determine BPM</i>	<i>7 Days</i>	3/9/11	3/16/11	Hard	Marvin
<i>Song Choice Algorithm</i>	<i>14 Days</i>	3/17/11	4/5/11	Hard	Marvin
<i>Tempo Adjustment Algorithm</i>	<i>14 Days</i>	3/17/11	4/5/11	Milestone	Team
Integration	15 Days	4/5/11	4/20/11		
<i>Combine all Modules</i>	<i>7 Days</i>	4/6/11	4/13/11	Milestone	Team
Test & Debug	6 Days	4/14/11	4/21/11		
<i>Fix Errors & Simulation</i>	<i>6 Days</i>	4/14/11	4/21/11	Hard	Team
Final Presentation	8 Days	4/20/11	4/29/11		
<i>Prepare Presentation</i>	<i>8 Days</i>	4/20/11	4/29/11	Medium	Team
<i>Present</i>	<i>0 Days</i>	4/29/11	4/29/11	Milestone	Team
Final Demonstration	5 Days	4/29/11	4/5/11		
<i>Prepare Demonstration</i>	<i>5 Days</i>	4/29/11	4/5/11	Medium	Team
<i>Give Demo</i>	<i>0 Days</i>	5/5/11	5/5/11	Milestone	Team
Final Report	0 Days	5/5/11	5/5/11		